Forward Thinkers: Final Presentation

WILLIAM YE (Z5061340)

JIAHAO GE (Z5211410)

DANIEL ROSENGARTEN (Z5160057)

Presentation Outline

Project Aims

Challenges

Architecture

Dense Layer Acceleration - **William**

Convolutional Layer Acceleration – **Jiahao**

Results and Discussion - **Daniel**

Project Aims

- Focus on the speedup of two neural network layers:
  - Dense Layer
  - Convolutional Layer
- Bonus Objectives:
  - Activation Layer
  - Pooling Layer
  - Batch Norm / Dropout
  - Complete CNN network

Challenges

- Short turnaround time for the project
- Managing time and organizing meetings
- Time zone differences

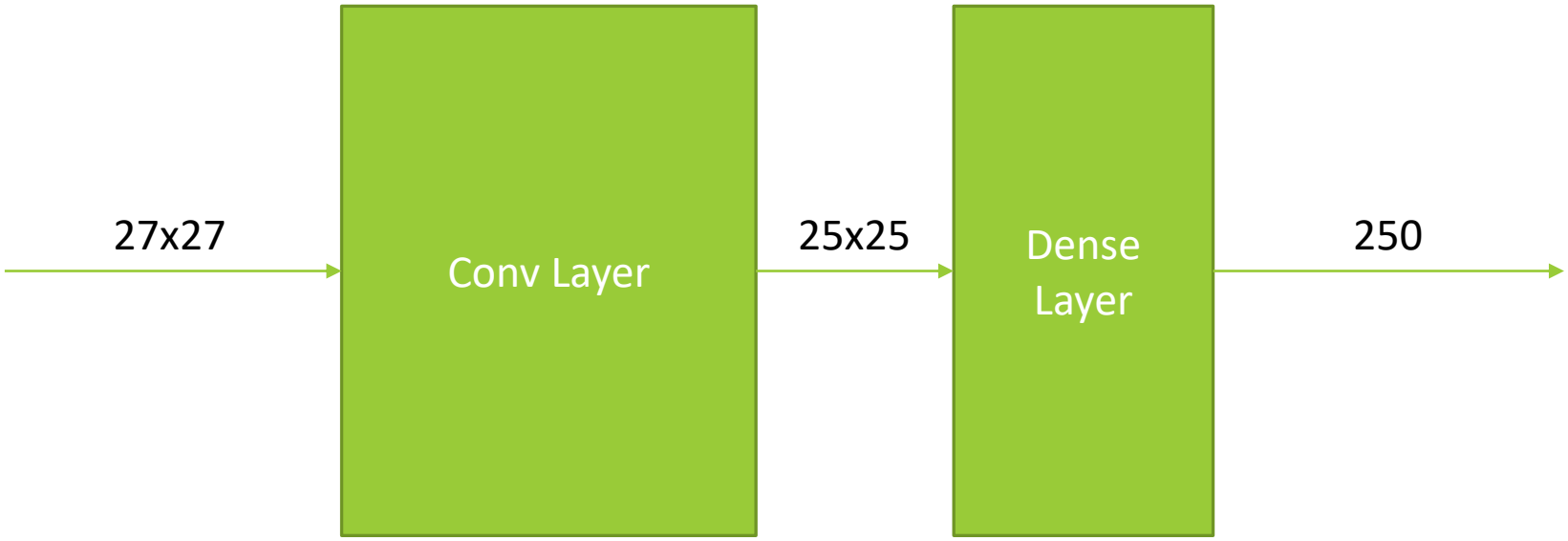
Design Approach

Divided the work into acceleration of:

- Dense Layer - William
- Convolutional Layer - Daniel & Jiahao

Architecture

- Convolutional layer included input size of 27x27, output size of 25x25, filter 3x3
- Dense layer included input size 625 and output size of 250
- Number of Batches kept at 1
- xczu7ev-ffvc1156-2-e device used



Dense Layer – Baseline Solution

```c
#include "dense.h"

void dense_layer(
        double inputs[BATCH_SIZE][INPUT_SIZE],
        double weights[OUTPUT_SIZE][INPUT_SIZE],
        double bias[OUTPUT_SIZE],
        double outputs[BATCH_SIZE][OUTPUT_SIZE])
{
    int batch, i, j;

    batch_loop: for (batch = 0; batch < BATCH_SIZE; batch++) {
        // For each batch, determine the output layer produced by that batch

        output_neurons: for (i = 0; i < OUTPUT_SIZE; i++) {
            // For each output neuron, the output is the sum of all input
            // neurons multiplied by their respective weights, plus a bias
            // output = inputs * weights + bias

            double neuron_output = 0.0;
            mac: for (j = 0; j < INPUT_SIZE; j++) {
                neuron_output += inputs[batch][j] * weights[i][j];
            }
            outputs[batch][i] = neuron_output + bias[i];
        }
    }
}
```

```
real    0m0.029s
user    0m0.000s
sys     0m0.000s
```

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.423 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1720751	1720751	17.208 ms	17.208 ms	1720751	1720751	none

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	106	-
FIFO	-	-	-	-	-
Instance	-	14	729	965	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	148	-
Register	-	-	491	-	-
Total	0	14	1220	1219	0
Available	1590	1260	728400	364200	0
Utilization (%)	0	1	~0	~0	0

Unrolling the Dense Layer

Latency

		solution1	solution4_unroll2	solution5_unroll4	solution6_unroll6	solution7_unroll_comp
Latency (cycles)	min	1720751	1720626	1734342	1734335	1720500
	max	1720751	1720626	1734342	1734335	1720500
Latency (absolute)	min	17.208 ms	17.206 ms	17.343 ms	17.343 ms	17.683 ms
	max	17.208 ms	17.206 ms	17.343 ms	17.343 ms	17.683 ms
Interval (cycles)	min	1720751	1720626	1734342	1734335	1720500
	max	1720751	1720626	1734342	1734335	1720500

Utilization Estimates

	solution1	solution4_unroll2	solution5_unroll4	solution6_unroll6	solution7_unroll_comp
BRAM_18K	0	0	0	0	0
DSP48E	14	16	18	20	14
FF	1220	1335	1747	2073	26300
LUT	1219	1409	1890	2329	26162
URAM	0	0	0	0	0

Outer Loop

Latency

		solution1	solution8_unroll2_inner	solution9_unroll4_inner	solution10_unroll_comp_inner
Latency (cycles)	min	1720751	1642501	1135501	628001
	max	1720751	1642501	1135501	628001
Latency (absolute)	min	17.208 ms	16.425 ms	11.355 ms	6.417 ms
	max	17.208 ms	16.425 ms	11.355 ms	6.417 ms
Interval (cycles)	min	1720751	1642501	1135501	628001
	max	1720751	1642501	1135501	628001

Utilization Estimates

	solution1	solution8_unroll2_inner	solution9_unroll4_inner	solution10_unroll_comp_inner
BRAM_18K	0	0	0	0
DSP48E	14	14	14	14
FF	1220	1230	1366	3750
LUT	1219	1339	1495	23279
URAM	0	0	0	0

MAC Loop

Pipelining the Dense Layer

Latency

		solution1	solution2_pipinner	solution3_pipouter
Latency (cycles)	min	1720751	781263	628001
	max	1720751	781263	628001
Latency (absolute)	min	17.208 ms	7.813 ms	6.417 ms
	max	17.208 ms	7.813 ms	6.417 ms
Interval (cycles)	min	1720751	781263	628001
	max	1720751	781263	628001

Utilization Estimates

	solution1	solution2_pipinner	solution3_pipouter
BRAM_18K	0	0	0
DSP48E	14	15	14
FF	1220	1498	3750
LUT	1219	1366	23279
URAM	0	0	0

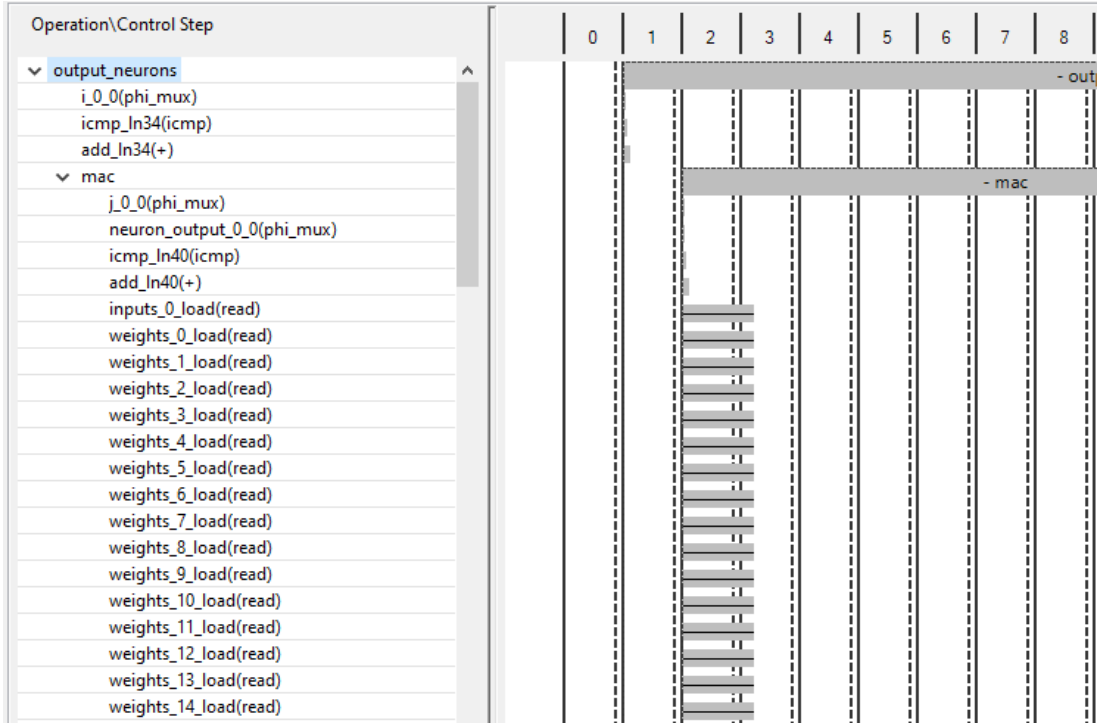
Array Partitioning

Latency

		solution1	solution13_arraypartition
Latency (cycles)	min	1720751	1720501
	max	1720751	1720501
Latency (absolute)	min	17.208 ms	17.205 ms
	max	17.208 ms	17.205 ms
Interval (cycles)	min	1720751	1720501
	max	1720751	1720501

Utilization Estimates

	solution1	solution13_arraypartition
BRAM_18K	0	0
DSP48E	14	14
FF	1220	1175
LUT	1219	3820
URAM	0	0



Fixed Point for Dense Layer

Latency

		solution1	solution2_apfixed
Latency (cycles)	min	1720751	313251
	max	1720751	313251
Latency (absolute)	min	17.208 ms	3.133 ms
	max	17.208 ms	3.133 ms
Interval (cycles)	min	1720751	313251
	max	1720751	313251

Utilization Estimates

	solution1	solution2_apfixed
BRAM_18K	0	0
DSP48E	14	1
FF	1220	103
LUT	1219	192
URAM	0	0

```
typedef ap_fixed<10,1> DATA_TYPE;
```

```
#include "dense.h"

void dense_layer(
        DATA_TYPE inputs[BATCH_SIZE][INPUT_SIZE],
        DATA_TYPE weights[OUTPUT_SIZE][INPUT_SIZE],
        DATA_TYPE bias[OUTPUT_SIZE],
        DATA_TYPE outputs[BATCH_SIZE][OUTPUT_SIZE])
{
    int batch, i, j;

    batch_loop: for (batch = 0; batch < BATCH_SIZE; batch++) {
        // For each batch, determine the output layer produced by that batch

        output_neurons: for (i = 0; i < OUTPUT_SIZE; i++) {
            // For each output neuron, the output is the sum of all input
            // neurons multiplied by their respective weights, plus a bias
            // output = inputs * weights + bias

            DATA_TYPE neuron_output = 0.0;
            mac: for (j = 0; j < INPUT_SIZE; j++) {
                neuron_output += inputs[batch][j] * weights[i][j];
            }
            outputs[batch][i] = neuron_output + bias[i];
        }
    }
}
```

Final Solution - Dense Layer

Latency

		solution1	final_solution
Latency (cycles)	min	1720751	78751
	max	1720751	78751
Latency (absolute)	min	17.208 ms	0.788 ms
	max	17.208 ms	0.788 ms
Interval (cycles)	min	1720751	78751
	max	1720751	78751

Utilization Estimates

	solution1	final_solution
BRAM_18K	0	0
DSP48E	14	624
FF	1220	3526
LUT	1219	22758
URAM	0	0

- Pipelining the Inner Loop
- Unrolling the Inner Loop
- Fixed Point Data Types

Convolutional Layer - Solution 1 (baseline)

Without any directives

```
#include "conv.h"

void conv_layer(
        double inputs[BATCH_SIZE][INPUT_HEIGHT][INPUT_WIDTH],
        double filter[FILT_HEIGHT][FILT_WIDTH],
        double outputs[BATCH_SIZE][OUTPUT_HEIGHT][OUTPUT_WIDTH])
{
    int batch, out_i, out_j, filt_i, filt_j;
    int stride = STRIDE;
    for (batch = 0; batch < BATCH_SIZE; batch++) {
        // For every batch calculate the output layer
        for (out_i = 0; out_i < OUTPUT_HEIGHT; out_i++) {
            for (out_j = 0; out_j < OUTPUT_WIDTH; out_j++) {
                // For every output neuron, convolve the input with the filter
                double conv = 0.0;
                for (filt_i = 0; filt_i < FILT_HEIGHT; filt_i++) {
                    for (filt_j = 0; filt_j < FILT_WIDTH; filt_j++) {
                        // The starting location of the input depends on the stride
                        int in_i = out_i * stride + filt_i;
                        int in_j = out_i * stride + filt_j;
                        conv += inputs[batch][in_i][in_j] * filter[filt_i][filt_j];
                    }
                }
                outputs[batch][out_i][out_j] = conv;
            }
        }
    }
}
```

```
real    0m0.030s
user    0m0.000s
sys     0m0.000s
```

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.279 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
66926	66926	0.669 ms	0.669 ms	66926	66926	none

Detail

Instance

Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	66925	66925	2677	-	-	25	no
+ Loop 1.1	2675	2675	107	-	-	25	no
++ Loop 1.1.1	105	105	35	-	-	3	no
+++ Loop 1.1.1.1	33	33	11	-	-	3	no

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	0	0	214	-
FIFO	-	-	-	-	-
Instance	-	14	744	985	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	128	-
Register	-	-	270	-	-
Total	0	14	1014	1327	0
Available	624	1728	460800	230400	96
Utilization (%)	0	~0	~0	~0	0

Convolutional Layer-Solution 2

Change the datatype

```
#ifndef _test_
#define _test_

#include "ap_fixed.h"

typedef ap_fixed<10,1>  DATA_TYPE;
typedef ap_fixed<20,2>  DATA_TYPE_O;
```

```
#include "conv.h"

void conv_layer(
        DATA_TYPE inputs[BATCH_SIZE][INPUT_HEIGHT][INPUT_WIDTH],
        DATA_TYPE filter[FILT_HEIGHT][FILT_WIDTH],
        DATA_TYPE_O outputs[BATCH_SIZE][OUTPUT_HEIGHT][OUTPUT_WIDTH])
```

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	3.770 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
16301	16301	0.163 ms	0.163 ms	16301	16301	none

Detail

Instance

Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- output_i	16300	16300	652	-	-	25	no
+ output_j	650	650	26	-	-	25	no
++ filt_i	24	24	8	-	-	3	no
+++ filt_j	6	6	2	-	-	3	no

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	1	-	-	-
Expression	-	0	0	214	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	101	-
Register	-	-	109	-	-
Total	0	1	109	315	0
Available	624	1728	460800	230400	96
Utilization (%)	0	~0	~0	~0	0

Convolutional Layer-Solution 3

set the pipeline directive

```
1  ############################################################
2  ## This file is generated automatically by Vivado HLS.
3  ## Please DO NOT edit it.
4  ## Copyright (C) 1986-2020 Xilinx, Inc. All Rights Reserved.
5  ############################################################
6  set_directive_pipeline "conv_layer/conv_layer_label0"
7  set_directive_pipeline "conv_layer/conv_layer_label0"
8
```

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	5.286 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
7501	7501	75.010 us	75.010 us	7501	7501	none

Detail

Instance

Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- output_i_output_j	7500	7500	12	-	-	625	no
+ filt_i_conv_layer_label0	9	9	2	1	1	9	yes

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	2	-	-	-
Expression	-	0	0	220	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	114	-
Register	-	-	87	-	-
Total	0	2	87	334	0
Available	624	1728	460800	230400	96
Utilization (%)	0	~0	~0	~0	0

Convolutional Layer-Solution 3 (alt)

Unroll lower loops

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.678 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
3801	3801	38.010 us	38.010 us	3801	3801	none

Detail

Instance

Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- output_i	3800	3800	152	-	-	25	no
+ output_j	150	150	6	-	-	25	no

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	0	0	1037	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	200	-
Register	-	-	258	-	-
Total	0	0	258	1237	0
Available	624	1728	460800	230400	96
Utilization (%)	0	0	~0	~0	0

Convolutional Layer- Solution 4

```
void conv_layer(
        DATA_TYPE inputs[BATCH_SIZE][INPUT_HEIGHT][INPUT_WIDTH],
        DATA_TYPE filter[FILT_HEIGHT][FILT_WIDTH],
        DATA_TYPE_O outputs[BATCH_SIZE][OUTPUT_HEIGHT][OUTPUT_WIDTH])
{
    int batch, out_i, out_j, filt_i, filt_j;
    int stride = STRIDE;
    DATA_TYPE_O conv = 0.0;
    batch : for (batch = 0; batch < BATCH_SIZE; batch++) {
        // For every batch calculate the output layer
        output_i : for (out_i = 0; out_i < OUTPUT_HEIGHT; out_i++) {
           output_j : for (out_j = 0; out_j < OUTPUT_WIDTH; out_j++) {
                // For every output neuron, convolve the input with the filter
                //DATA_TYPE_O conv = 0.0;
                filt_i: for (filt_i = 0; filt_i < FILT_HEIGHT; filt_i++) {
                    filt_j : conv_layer_label0:for (filt_j = 0; filt_j < FILT_WIDTH; filt_j++) {
                        if((filt_j==0)&&(filt_i==0)) conv = 0;
                        // The starting location of the input depends on the stride
                        int in_i = out_i * stride + filt_i;
                        int in_j = out_i * stride + filt_j;
                        conv += inputs[batch][in_i][in_j] * filter[filt_i][filt_j];
                        if((filt_j==2)&&(filt_i==2)) outputs[batch][out_i][out_j] = conv;
                    }
                }
            }
        }
    }
}
```

Change the loop to a Perfect Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	3	-	-	-
Expression	-	-	0	339	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	135	-
Register	-	-	80	-	-
Total	0	3	80	474	0
Available	624	1728	460800	230400	96
Utilization (%)	0	~0	~0	~0	0

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	7.442 ns	1.25 ns

Latency

Summary

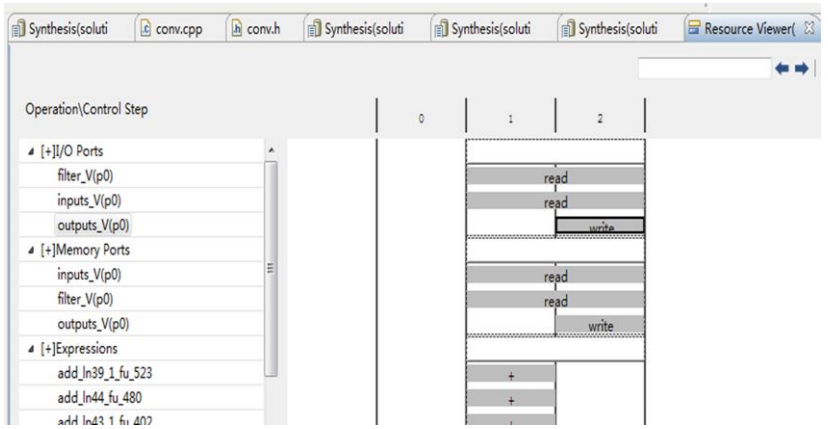
Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
5627	5627	56.270 us	56.270 us	5627	5627	none

Detail

Instance

Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- output_i_filt_i_filt_j	5625	5625	2	1	1	5625	yes



Resource and Flow

Convolutional Layer-Solution 5

continue to change the directive

```
1  ############################################################
2  ## This file is generated automatically by Vivado HLS.
3  ## Please DO NOT edit it.
4  ## Copyright (C) 1986-2020 Xilinx, Inc. All Rights Reserved.
5  ############################################################
6  set_directive_pipeline "conv_layer/conv_layer_label2"
7  set_directive_pipeline "conv_layer/output_j"
8  set_directive_unroll "conv_layer/filt_i"
9  set_directive_unroll "conv_layer/filt_j"
10
```

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	6	-	-	-
Expression	-	0	0	639	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	224	-
Register	-	-	296	-	-
Total	0	6	296	863	0
Available	624	1728	460800	230400	96
Utilization (%)	0	~0	~0	~0	0

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.280 ns	1.25 ns

Latency

Summary

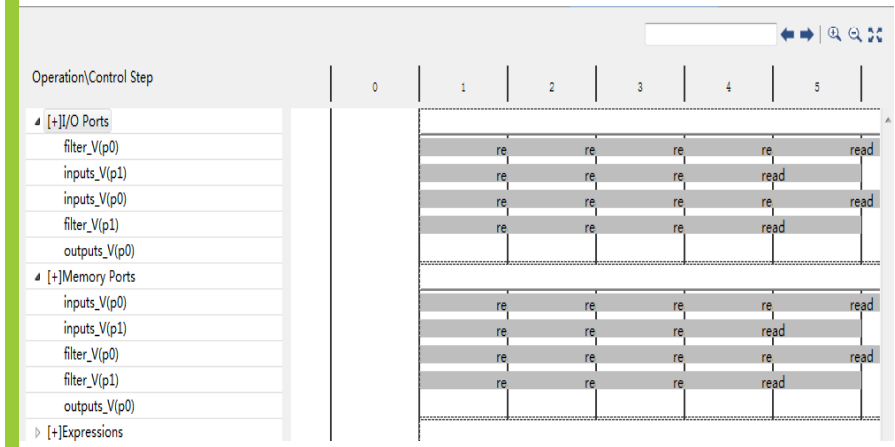
Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
3128	3128	31.280 us	31.280 us	3128	3128	none

Detail

Instance

Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- output_i_output_j	3126	3126	7	5	1	625	yes



Resource and Flow

Convolutional Layer-Solution 6

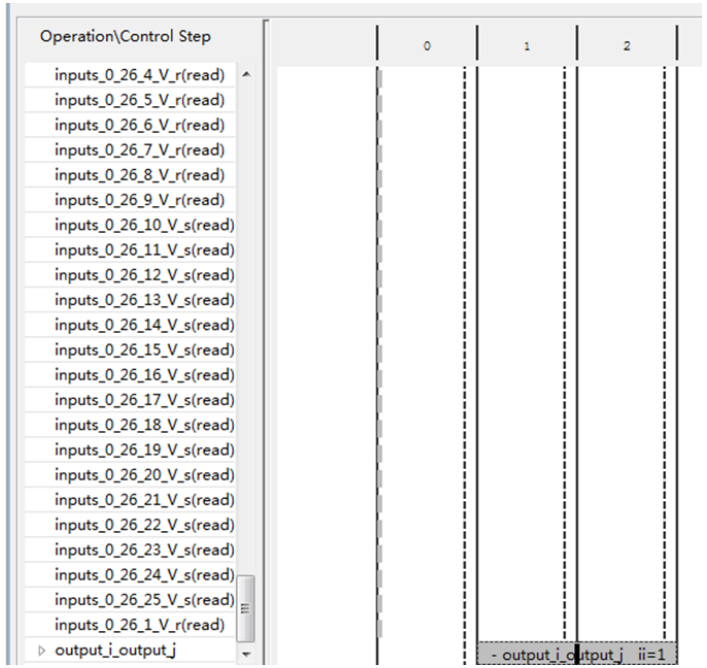
Optimize the RAM

```
############################################################
## This file is generated automatically by Vivado HLS.
## Please DO NOT edit it.
## Copyright (C) 1986-2020 Xilinx, Inc. All Rights Reserved.
############################################################
set_directive_pipeline "conv_layer/conv_layer_label2"
set_directive_pipeline "conv_layer/output_j"
set_directive_unroll "conv_layer/filt_j"
set_directive_array_partition -type complete -dim 0 "conv_layer" outputs
set_directive_array_partition -type complete -dim 0 "conv_layer" filter
set_directive_array_partition -type complete -dim 0 "conv_layer" inputs

```

```
#include "conv.h"

void conv_layer(
        DATA_TYPE inputs[BATCH_SIZE][INPUT_HEIGHT][INPUT_WIDTH],
        DATA_TYPE filter[FILT_HEIGHT][FILT_WIDTH],
        DATA_TYPE_O outputs[BATCH_SIZE][OUTPUT_HEIGHT][OUTPUT_WIDTH])
{
    int batch, out_i, out_j, filt_i, filt_j;
    int stride = STRIDE;
    DATA_TYPE_O conv = 0.0;
    batch : for (batch = 0; batch < BATCH_SIZE; batch++) {
        // For every batch calculate the output layer
        output_i : for (out_i = 0; out_i < OUTPUT_HEIGHT; out_i++) {
           output_j : for (out_j = 0; out_j < OUTPUT_WIDTH; out_j++) {
                 // For every output neuron, convolve the input with the filter
                //DATA_TYPE_O conv = 0.0;
                filt_i: for (filt_i = 0; filt_i < FILT_HEIGHT; filt_i++) {
                     filt_j : for (filt_j = 0; filt_j < FILT_WIDTH; filt_j++) {
                         //if((filt_j==0)&&(filt_i==0)) conv = 0;
                         // The starting location of the input depends on the stride
                         //int in_i = out_i * stride + filt_i;
                         //int in_j = out_i * stride + filt_j;
                         outputs[batch][out_i][out_j] += inputs[batch][out_i * stride + filt_i][out_i * stride + filt_j] * filter[filt_i][filt_j];
                         //if((filt_j==2)&&(filt_i==2)) outputs[batch][out_i][out_j] = conv;
                     }
                 }
                //outputs[batch][out_i][out_j] = conv;
            }
        }
    }
}
```



Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.439 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
627	627	6.270 us	6.270 us	627	627	none

Detail

Instance

Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- output_i_output_j	625	625	2	1	1	625	yes

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	1	-	-	-
Expression	-	0	0	793	-
FIFO	-	-	-	-	-
Instance	-	-	0	2693	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	7554	-
Register	-	-	216	-	-
Total	0	1	216	11040	0
Available	624	1728	460800	230400	96
Utilization (%)	0	~0	~0	4	0

Resource and Flow

Results – Dense Layer

- Speedup to 0.788ms (21.8x faster)
- Hardware usage greatly increased
- (DSP usage now over 1/3rd, others not overly utilised)
- Not viable for much greater input sizes
  - Could manually unroll with smaller factor to avoid large hardware utilisation
  - Could also use simple ap_fixed version without pipelining/unrolling for speedup without large data utilisation
- Batch sizes > 1 linearly increase latency

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	624	-	-	-
Expression	-	0	0	15791	-
FIFO	-	-	-	-	-
Instance	-	-	0	94	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	32231	-
Register	-	-	3563	-	-
Total	0	624	3563	48116	0
Available	1590	1260	728400	364200	0
Utilization (%)	0	49	~0	13	0

Latency

		solution1	final_solution
Latency (cycles)	min	1720751	78751
	max	1720751	78751
Latency (absolute)	min	17.208 ms	0.788 ms
	max	17.208 ms	0.788 ms
Interval (cycles)	min	1720751	78751
	max	1720751	78751

Utilization Estimates

	solution1	final_solution
BRAM_18K	0	0
DSP48E	14	624
FF	1220	3526
LUT	1219	22758
URAM	0	0

Results – Convolutional Layer

- Speedup to 6.27us (speedup of 52.8x)
- Final hardware usage not overly costly (10x LUTs)
- Batch size > 1 introduces linear speedup, but large increase in LUT usage
  - Might need to use less hardware intensive setup (solution 4)
- Same is true for larger inputs

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	0	0	19692	-
FIFO	-	-	-	-	-
Instance	-	-	0	5115	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	5871	-
Register	-	-	7054	-	-
Total	0	0	7054	30678	0
Available	624	1728	460800	230400	96
Utilization (%)	0	0	1	13	0

Timing

Clock		solution1	solution2	solution3	solution4	solution6
ap_clk	Target	10.00 ns	10.00 ns	10.00 ns	10.00 ns	10.00 ns
	Estimated	8.279 ns	3.770 ns	5.286 ns	7.442 ns	6.439 ns

Latency

		solution1	solution2	solution3	solution4	solution6
Latency (cycles)	min	33126	16301	7501	5627	627
	max	33126	16301	7501	5627	627
Latency (absolute)	min	0.331 ms	0.163 ms	75.010 us	56.270 us	6.270 us
	max	0.331 ms	0.163 ms	75.010 us	56.270 us	6.270 us
Interval (cycles)	min	33126	16301	7501	5627	627
	max	33126	16301	7501	5627	627

Utilization Estimates

	solution1	solution2	solution3	solution4	solution6
BRAM_18K	0	0	0	0	0
DSP48E	15	1	2	3	1
FF	953	109	87	80	216
LUT	1346	315	334	474	11040
URAM	0	0	0	0	0

What worked

- Focusing on only 2 layers
- Assigning more members to more complex convolutional layer
- Scheduling weekly meetings
- Using GitHub for code sharing

What Didn't

- Too much focus on acceleration, not enough focus on evaluation of new solutions
- Time management

Further Research

- Generating full CNN for actual MNIST dataset operations
- Accelerations for max pooling and dropout layers and activation functions
- More testing on very big inputs/filter size
- Completely training a neural network on an FPGA

Thank You For Listening!

ANY QUESTIONS?